(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 052 814 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
15.11.2000 Bulletin 2000/46

(51) Int. Cl.[7]: **H04L 12/56**, H04Q 11/04

(21) Application number: **00304076.3**

(22) Date of filing: **15.05.2000**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **14.05.1999 US 311833**

(71) Applicant:
Nortel Networks Limited
Montreal, Quebec H2Y 3Y4 (CA)

(72) Inventors:
- **Angle, Richard L.**
  Wellesley Hills, Massachusetts 02481 (US)
- **Jagannath, Shantigram V**
  Cambridge, Massachusetts 02139 (US)
- **Ladwig, Geoffrey B.**
  Chelmsford, Massachusetts 01824 (US)

(74) Representative:
Dearling, Bruce Clive et al
Hepworth Lawrence Bryer & Bizley,
Merlin House,
Falconry Court,
Bakers Lane
Epping, Essex CM16 5DQ (GB)

(54) Multicast scheduling for a network device

(57) A method and apparatus are provided for scheduling multicast data in an input-queued network device. According to one aspect of the present invention, deterministic and bounded delay for high priority multicast cells is guaranteed by the multicast scheduler. The scheduler receives a transmit request associated with each of a plurality of input ports. The transmit request identifies output ports to which pending multicast cells are ready to be transmitted, if any. Then, for each of multiple classes of service, the scheduler performs a single scheduling iteration. The single scheduling iteration includes a grant phase, an accept phase, and an update phase. During the grant phase, the scheduler grants one or more of the input ports access to the fabric by issuing grants based upon the transmit requests and a priority indicator that identifies an input port that is given scheduling priority for the scheduling iteration. During the accept phase, on behalf of each of the input ports, the scheduler accepts all grants corresponding to the input port. Finally, during the update phase, the scheduler updates the priority indicator for use in a subsequent scheduling cycle.

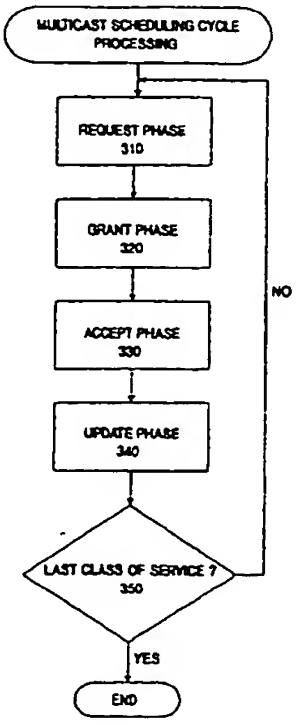

FIG. 3

EP 1 052 814 A2

numerals refer to similar elements and in which:

Figure 1 is a simplified block diagram of a network device according to one embodiment of the present invention.

Figure 2 is a high level block diagram of various functional units that may be employed in a fabric configuration manager according to one embodiment of the present invention.

Figure 3 is a flow diagram illustrating multicast scheduling processing according to one embodiment of the present invention.

Figure 4 conceptually illustrates a multicast scheduling cycle according to one embodiment of the present invention.

Figure 5 is a high level block diagram of a hardware implementation of a multicast scheduler according to one embodiment of the present invention.

Figure 6 is a flow diagram illustrating unicast scheduling processing according to one embodiment of the present invention.

Figures 7A and 7B conceptually illustrate a unicast scheduling cycle according to one embodiment of the present invention.

Figure 8 is a high level block diagram of a hardware implementation of a unicast scheduler according to one embodiment of the present invention.

Figures 9A and 9B depict exemplary round-robin arbiters that may be used in accordance with one embodiment of the present invention.

Figure 10A is a flow diagram illustrating combined scheduling processing for two types of traffic according to one embodiment of the present invention.

Figure 10B is a flow diagram illustrating combined unicast and multicast scheduling processing according to one embodiment of the present invention.

Figure 10C is a flow diagram illustrating combined unicast and multicast scheduling processing according to another embodiment of the present invention.

Figure 11A conceptually illustrates a pipelined approach for scheduling multicast and unicast traffic according to one embodiment of the present invention.

Figure 11B conceptually illustrates a pipelined approach for scheduling multicast and unicast traffic according to one embodiment of the present invention.

Figure 12 is a flow diagram illustrating backpressure processing according to one embodiment of the present invention.

Figure 13 is a block diagram of a hardware implementation of a portion of the backpressure logic according to one embodiment of the present invention.

Detailed Description Of The Invention

[0010] A method and apparatus are described for scheduling unicast and multicast traffic in an input-queued network device. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some of these specific details. In other instances, well-known structures and devices are shown in block diagram form.

[0011] The present invention includes various steps, which will be described below. The steps of the present invention may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause a general-purpose or special-purpose processor or logic circuits programmed with the instructions to perform the steps. Alternatively, the steps may be performed by a combination of hardware and software.

[0012] Assuming sufficient processing speed can be made available to accommodate cell scheduling time constraints, it is envisioned that the present invention may also be provided as a computer program product which may include a machine-readable medium having stored thereon instructions which may be used to program a computer (or other electronic devices) to perform a process according to the present invention. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, flash memory, magnet or optical cards, or other type of media / machine-readable medium suitable for storing electronic instructions. Moreover, the present invention may also be downloaded as a computer program product, wherein the program may be transferred from a remote computer (e.g., a server) to a requesting computer (e.g., a client) by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection).

[0013] Importantly, while embodiments of the present invention will be described with respect to a network device, such as a router, or a Layer 2 or Layer 3 switch operable within a TCP/IP network, the method and apparatus described herein are equally applicable to Asynchronous Transfer Mode (ATM) networks and networking devices, and other devices such as multiprocessing computers, for example. Additionally, while various embodiments of the present invention are described in connection with a networking device that recognizes four different classes of service, the method and apparatus described herein are not limited to such a configuration.

215 and takes advantage of the multicast capability of the fabric 120. Details regarding the implementation of the multicast scheduler 215 and the processing performed by the multicast scheduler 215 are described below.

[0023] The unicast scheduler 220 implements a fabric arbitration algorithm for unicast cells. In one embodiment, the unicast scheduler 220 operates in accordance with the iSLIP unicast scheduling algorithm which is described below. However, in other embodiments, a variety of other unicast scheduling approaches may be employed such as, Programmable Iterative Matching (PIM), iterative Longest Queue First (i-LQF), iterative Oldest Cell First (i-OCF) or variants and combinations thereof.

[0024] The time slot scheduling control logic 210 initiates multicast and unicast scheduling during the appropriate scheduling time slots. In one embodiment, unicast scheduling is performed every time slot while multicast scheduling is performed every other time slot. In another embodiment, the frequency at which multicast scheduling is performed is a programmable parameter, e.g., multicast scheduling frequency 245, that may be provided by the network administrator, for example. Assuming that a time slot is not long enough to allow both multicast scheduling and unicast scheduling to be completed if performed sequentially, according to one embodiment, the scheduling of multicast and unicast cells that are to be delivered during the same time slot may be staged in a pipelined fashion. For example, multicast scheduling may be performed in advance of the time slot in which the results are used. In this manner, during time slots in which only unicast cells are transferred across the fabric 120, both multicast and unicast scheduling may be performed independently and in parallel. During time slots in which both multicast and unicast cells are to be transferred, the results of the multicast scheduling cycle that were performed in advance are fed into the unicast scheduler 220 and the unicast scheduler 220 then schedules unicast cells whose ports are not being used by the previously scheduled multicast cells. Importantly, while specific multicast and unicast scheduling approaches are described herein, the mechanism for producing a combined schedule is not limited to these particular approaches. The novel separation and pipelined staging of multicast and unicast scheduling and the parallel operation of the multicast and unicast scheduling that will be described further below are equally applicable to other current and future scheduling approaches.

[0025] In the embodiment depicted, the fabric configuration manager 110 has control information interfaces with the input ports 107, the output ports 109, and the fabric 120.

[0026] Control information generated by the fabric configuration manager 110 includes information regarding queue selection 240 and 255 which is sent each time slot to those of the input ports and output ports participating in the schedule generated by the multicast scheduler 215 and/or the unicast scheduler 220. Additionally, each time slot, the fabric configuration manager 110 produces a fabric configuration 260 based upon the current schedule. The fabric configuration 260 is communicated to the fabric 120 each time slot to activate the fabric 120 and cause the fabric 120 to form appropriate connections among the fabric interfaces 115 to accommodate the current schedule.

[0027] In this example, control information received by the fabric configuration manager 110 from the input ports 107 includes input queue state information 230 and transmit requests 235. According to one embodiment, input queue state information 230 comprises information on newly received cells such as an indication of the queue with which the cell is associated (identified by the port and the class of service, for example) and the output port(s) to which the cell is destined. In this manner, the current state of the queues at each of the input ports may be maintained in the input queue status block 205. In alternative embodiments, queue status may be kept at the input ports 107.

[0028] Transmit requests 235 may be received from the input ports 107 at the beginning of each time slot. The transmit requests 235 identify the output port(s), if any, to which the corresponding input port has a cell ready to be transferred. As will be described further below, the transmit requests 235 may be presented to one or both of the multicast scheduler 215 and the unicast scheduler 220 in the form of request vectors for each output port 109. Each request vector identifies those of the input ports 107 with requests for a particular output port 109.

[0029] The fabric configuration manager 110 also receives control information from the output ports 109. For example, a back pressure signal 250 may identify output ports 109 having one or more output queues that have exceeded a predetermined threshold of pending cells. Briefly, in network devices employing "speedup," e.g., those operating their fabrics faster than the input and output port capacity, a back pressure signaling mechanism is important to protect the output ports from excess traffic from the fabric. In prior architectures, a back pressure signal is typically coupled directly from each of the output ports 109 to each of the input ports 107. Typically, output ports assert their back pressure signal upon exceeding a predetermined threshold of pending cells. Subsequently, when the number of pending cells falls below another predetermined threshold, the output port deasserts the back pressure signal. While an output port is back pressuring, input ports do not transfer cells to that output port. When VOQs are employed at the input ports 107, back pressure does not cause difficulties for unicast traffic since a head-of-line unicast cell destined for a back pressuring output only blocks other cells destined for the same output. However, a complication occurs for multicast traffic. When a particular output port is back pressuring, a

one embodiment, the request vectors may be modified based upon the accepts such that no requests will be presented in subsequent iterations from an input port that has accepted an output port and/or such that no requests will be presented to an output port that has been accepted by an input port.

[0038] At step 350, it is determined if the last class of service has been scheduled. If not, processing continues with step 310 for the next class of service. Otherwise, if the last class of service has been scheduled, then the multicast scheduling cycle is complete.

[0039] Referring now to Figure 4, one iteration of an exemplary multicast scheduling cycle will be described. Communication of transmit request information is depicted as a solid line from an input port to an output port and grants are depicted as dotted lines from output ports to input ports. According to this example, there are four input ports numbered 0 to 3 and four output ports numbered 0 to 3. The input ports each have a corresponding unavailability indicator 405-408. Similarly, each output port has a corresponding unavailability indicator 415-418. In this example, the global priority indicator comprises a GRRC 450 which currently points to input port 1. Therefore, input port 1 will receive priority over the other input ports during this scheduling cycle.

[0040] In this example, input port 0 has a multicast cell ready for delivery to output ports 1 and 2, the head-of-line multicast cell at input port 1 is destined for output ports 0, 1 and 3, no multicast cells are pending at input port 2, and input port 3 has a multicast cell that is ready to be transferred to output port 2. While, in reality, this control information is communicated from the input ports 107 to the fabric configuration manager 110, for purposes of this example, the communication is conceptually represented as being communicated between input ports 0-3 and output ports 0-3. At any rate, during the request phase (step 310) the input ports each communicate their transmit request information to the output ports.

[0041] The transmit request information accumulated at output port 0 forms request vector 425. Similarly, request vectors 426-428 are formed at output ports 1-3, respectively. The request vectors 425-428 have a bit position corresponding to each of the input ports 0-3. A bit is set in the request vectors 435-428 when the corresponding input port has a multicast cell ready for delivery to the output port. The bit remains clear if the corresponding input port has no cell destined for the output port.

[0042] During the grant phase (step 320), each output port that is available searches its corresponding request vector 425-428, respectively, in a circular fashion beginning with the input port identified by GRRC 450 to select the first available input port 0-3 that has a request for that output port. In this example, the first available input port that has a request for output port 0 is input port 1. Input port 1 is also the first available input port that has a request for output port 1. Output port 2 selects input port 3 and output port 3 selects input port 1. After the output ports have selected an input port, grants are issued to those of the input ports that have been selected and those of the output ports that gave a grant to an input port set their corresponding unavailability indicators. In this example, therefore, all of the output ports would set their output unavailability indicators 415-418.

[0043] During the accept phase (step 330), each input port accepts all grants received and each input port that received at least one grant sets its input unavailability indicator 405-408. In this example, input ports 1 and 3 would set their corresponding unavailability indicators. Therefore, input ports 1 and 3 would not participate in subsequent iterations until a new scheduling cycle begins.

[0044] During the update phase (step 340), the GRRC update criteria are evaluated and the GRRC 450 is incremented to point to input port 2. Therefore, in the next scheduling cycle, input port 2 will receive priority over input ports 0, 1 and 3.

[0045] Assuming the fabric 120 were configured according to this scheduling cycle iteration, the multicast cell from input port 3 would be delivered to output port 2 and the multicast cell from input port 1 would be delivered to output ports 0, 1 and 3.

[0046] Figure 5 is a high level block diagram of a hardware implementation of a multicast scheduler according to one embodiment of the present invention. According to this embodiment, the multicast scheduler 500 consists of a set of priority request vector registers for each class of service 510-513, a multiplexer (MUX) 520, iteration control logic 530, a set of active request vector registers 540, a set of output grant arbiters 550, a set of priority indicators, e.g., GRRCs 580-583, a MUX 560, and a set of grant vector registers 570.

[0047] During a multicast scheduling cycle, iteration control logic 530 causes an arbitration iteration to be performed for each class of service. The priority request vector registers 510-513 are coupled to the input of MUX 520. Iteration control logic 530 is coupled to MUX 520 to select the set of priority request vectors for the current iteration. The output of MUX 520 is coupled to the set of active request vector registers 540. As a result, the priority request vectors selected for the current iteration are passed to the active request vector registers 540. Iteration control logic 530 is also coupled to MUX 560 to select the GRRC appropriate for the current iteration. The GRRCs 580-583 are coupled to the input of MUX 560 and the output of MUX 560 is coupled to each of the output grant arbiters 55, thereby programming each output grant arbiter 550 with the selected GRRC.

[0048] The active request vector registers 540 are coupled to the output grant arbiters 550. Each of the output grant arbiters 550 are presented with priorities and a request vector from the active request vector registers 540. For example, output grant arbiter 0 receives

accepted. Based upon the results of the evaluation of the grant vectors, accepts are made to the selected outputs. Finally, those of the input ports 107 and output ports 109 that were matched during the current iteration (as determined by the accepts) should be marked as unavailable. In this manner, the matched ports are removed from further consideration in subsequent iterations of the scheduling cycle.

[0055] At step 640, the update phase is performed. The update phase includes updating the request vectors for use in the next iteration and updating the priority indicators, e.g., the ORRC values and the IRRC values. In one embodiment, the request vectors may be modified based upon the accepts such that no requests will be presented in subsequent iterations from an input port that has accepted an output port and/or such that no requests will be presented to an output port that has been accepted by an input port. With regard to update of the priority indicators, an ORRC is incremented if either the input port to which it points has been serviced by the corresponding output port (e.g., the output port has issued a grant to the input port which has been accepted for the corresponding class of service) or the input port requires no service from the output port (e.g., the input port has no unicast cells pending for the corresponding output port at the corresponding class of service). Similarly, an IRRC is incremented if either the output port to which it points has been serviced by the corresponding input port (e.g., the input port has accepted a grant to the output port for the corresponding class of service) or the output port requires no service from the input port (e.g., the output port has issued no grant to the corresponding input port for the corresponding class of service). Importantly, the priority indicators are updated only after the first iteration of a scheduling cycle.

[0056] Before discussing the update mechanism further, it may be useful to describe the overall functioning of the ORRCs and the IRRCs. The ORRCs and the IRRCs are tools that ensure that the unicast scheduling algorithm works in a fair manner and that the delays seen by all the input ports are finitely bounded. Each scheduling cycle, the ORRC for a particular output port steps through the request vector for the output port until it encounters a request (e.g., the bit it is pointing to is set indicating the corresponding input port has issued a request to the output port). The ORRC remains at this value until the request is serviced. This behavior ensures that the output port will always send a grant to this input port at every opportunity. Similarly, the IRRC for a particular input port steps through the corresponding grant vector until it encounters a grant (e.g., the bit it is pointing to is set indicating the corresponding output port has given a grant to the input port). The IRRC remains at this value until this grant is accepted. This behavior ensures that the input port will always accept the grant from this output port at every opportunity. Without such a gating mechanism, there is no guarantee that a cell will be transmitted. For example, if an ORRC was allowed to move ahead of a request without servicing it or if an IRRC was allowed to move ahead of a grant without servicing it, then the possibility exists that the corresponding cell might never be transmitted.

[0057] Returning to the updating of the ORRC and IRRC values, as was mentioned above, they are incremented only after the first iteration of a scheduling cycle. Additionally, the priority indicators may be updated differently depending upon the circumstances. For example, when a port to which a priority indicator points does not need service, the priority indicator is simply incremented by one (modulo N). However, when a port to which a priority indicator points needs service, the priority indicator is only updated after that port has been serviced; otherwise the priority indicator is not updated. The ORRC and IRRC values are incremented beyond a port that has just been serviced. For each IRRC, if the corresponding input port has accepted an output port for the corresponding class of service, then the IRRC is incremented to point to the output port after the one accepted. That is, the IRRC is set to one plus the value of the output port accepted (modulo N). Similarly, for each ORRC, if the corresponding output port has been accepted by an input port for the corresponding class of service, then the ORRC is set to one plus the value of the input port that has accepted the output port (modulo N). In this manner, connections made in the first iteration are the lowest priority during the next scheduling cycle and no connections are starved.

[0058] At step 650, it is determined if this is the last iteration of the scheduling cycle. For example, a predetermined number of iterations may be performed or the iterations may continue so long as at least one port matching was made during the previous iteration. In any event, the number of iterations need not exceed the number of input ports, N, since only one connection can be made for each Input port. If this is not the last iteration of the scheduling cycle, processing continues with step 610. Otherwise, the unicast scheduling cycle is complete and fabric configuration may commence at step 660.

[0059] At step 660, fabric configuration is performed. The fabric 120 is configured one time for each unicast scheduling cycle after the current schedule, e.g., the port matchings, has been established. Based upon the state of the accept signals at the conclusion of the unicast scheduling cycle, the unicast scheduler 220 presents a fabric configuration 260 to the fabric 120. The fabric configuration 260 activates the fabric 120 and directs the fabric 120 to form connections among the fabric interfaces 115 that will accommodate the transfer of cells across the fabric 120 according to the current schedule.

[0060] Referring to Figures 7A and 7B, one iteration of an exemplary unicast scheduling cycle will now be described. Beginning with Figure 7A, communication of transmit request information is depicted as a solid line

[0067] Figure 8 is a high level block diagram of a hardware implementation of a unicast scheduler according to one embodiment of the present invention. For purposes of simplifying the discussion, this example addresses the case of a unicast scheduler 800 that supports a single class of service. According to this embodiment, the unicast scheduler 800 consists of a set of active request vector registers 810, a set of output grant arbiters 850, a set of input accept arbiters 860, and a set of accept vector registers 870.

[0068] The active request vector registers 810 are coupled to the output grant arbiters 850. The output grant arbiters 850 choose among contenting requests on behalf of the corresponding output port 109. At the beginning of each unicast scheduling cycle, transmit requests 235 from the input ports 107 are loaded into the active request vector registers 810. Each iteration of the scheduling cycle, N-bit request vectors 811 are presented to each of the N corresponding output grant arbiters 850.

[0069] Each of the output grant arbiters 850 are coupled to each of the input accept arbiters 860. The output grant arbiters 850 each select one of the competing requests that is closest in priority to its ORRC and issue a grant signal to the input accept arbiter 860 corresponding to the selected request.

[0070] The input accept arbiters 860 are coupled to the set of accept vector registers 810 to identify the output port that has been matched with the corresponding input port. The input grant arbiters 860 each select one of the competing grants received from the output grant arbiters 850 that is closest in priority to its IRRC and issue an accept signal corresponding to the selected output port. The input accept arbiters 860 present accept signals in the form of an accept vector 871 to the set of accept vector registers 870. Each iteration, feedback from the accept vector registers 870 may be used to mask off requests corresponding to ports that have already been matched during the scheduling cycle. Alternatively, once a port has been matched, the corresponding arbiter 850 or 860 may be disabled in all further iterations of the scheduling cycle in order to prevent the arbiter from making additional matches. At any rate, the accept signals are accumulated in the set of accept vector registers 870 during each iteration of the unicast scheduling cycle and, as discussed above, are used at the end of the unicast scheduling cycle to configure the fabric 120.

Exemplary Round-Robin Arbiters

[0071] Figures 9A and 9B depict exemplary round-robin arbiters that may be used in accordance with one embodiment of the present invention. Referring first to Figure 9A, an output grant arbiter 950 that understands four classes of service is illustrated. In this example, the output grant arbiter 950 includes a grant priority filter 905 and a programmable priority encoder 920. A plurality of request vectors 904 associated with one or more priority levels are received by the grant priority filter 905. The grant priority filter 905 selects the request vector associated with the highest priority class of service and allows those requests 915 to be presented to the programmable priority encoder 920. As is well known, programmable priority encoders select as an output one of its inputs as determined by a supplied priority indication. In this example, the programmable priority encoder 920 grants one of the requests 915 based upon the highest priority 910, e.g., an ORRC, supplied by the grant priority filter 905. The priority levels and the grants produced by N output grant arbiters 950 are presented to an input accept arbiter 990 such as that illustrated in Figure 9B. In this example, the input accept arbiter 990 includes an accept priority filter 945 and a programmable priority encoder 960. The accept priority filter 945 outputs the grants 955 associated with the highest priority class of service level. The programmable priority encoder 960 accepts one of the grants 955 based upon the highest priority 951, e.g., an IRRC, supplied by the accept priority filter 945.

[0072] In one embodiment, the unicast scheduler 220 may include N output grant arbiters 950 and N input accept arbiters 990. Importantly, however, the present invention is not limited to any particular round-robin arbiter, the multicast scheduler 215 and unicast scheduler 220 may employ various other types of round-robin arbiters. For example, one of the fast programmable priority encoders described in P. Gupta and N. McKeown, "Design and Implementation of a Fast Crossbar Scheduler," Hot Interconnects VI, Stanford University, August 1998, which is hereby incorporated by reference, may be used.

Combined Multicast and Unicast Scheduling

[0073] According to one embodiment of the present invention, a novel scheduling approach permits unicast scheduling processing and multicast scheduling processing to operate in parallel and independently resulting in a combined schedule comprising both unicast and multicast cells. Parallelism may be achieved, for example, by pipeline staging of unicast and multicast scheduling. Pipelining unicast and multicast scheduling is advantageous, for example, if the duration of the cell scheduling cycle is insufficient to accommodate both unicast and multicast scheduling in serial. Prior scheduling approaches produce uniform schedules that are limited to either all unicast cells or all multicast cells. In contrast, the novel combined scheduling approach permits a combined schedule to be produced comprising both unicast and multicast cells. In this manner, both multicast and unicast cells may be transferred across the fabric 120 during the same time slot. Additionally, the separation of unicast and multicast scheduling results in increased flexibility and programmability of the rate at which unicast and/or multicast traffic is serviced.

for transfer across the fabric 120, multicast cells are scheduled based upon the results of an earlier unicast scheduling cycle. For example, multicast cells may be scheduled at a lower priority than the unicast cells by limiting the ports available to the multicast scheduler 215 to those ports left unmatched by the earlier multicast schedule.

[0082] At step 1045, a determination is made whether or not to perform unicast scheduling based upon a predetermined unicast scheduling frequency 1006. If unicast scheduling is to be performed during the current time slot, then processing continues with steps 1055 and 1060. Steps 1055 and 1060 are preferably performed by separate and independent schedulers, such as multicast scheduler 215 and unicast scheduler 220 discussed above, thereby providing the ability to perform the multicast and unicast scheduling in parallel. At step 1060, a unicast scheduling cycle is performed and the results are stored for a subsequent time slot. Concurrently, in step 1055 a multicast scheduling cycle is performed for the current time slot. If both unicast and multicast cells are to be transferred during the current time slot, then the prior unicast scheduling results 1007 are input into the multicast scheduling processing (step 1055) and the multicast scheduler 215 schedules multicast cells whose ports are not being used by the previously scheduled unicast cells. When both scheduling cycles have been finalized, the combined scheduling processing is complete.

[0083] Returning to step 1045, if unicast scheduling is not to be performed during the current time slot, then processing continues with step 1050. In step 1050, a multicast scheduling cycle is performed for the current time slot. Again, if both unicast and multicast cells are to be transferred during the current time slot, then the prior unicast scheduling results 1007 are input into the multicast scheduling processing (step 1050) and the multicast scheduler 215 schedules multicast cells whose ports are not being used by the previously scheduled unicast cells. When the scheduling cycle has been finalized, the combined scheduling processing is complete.

Pipelined Staging of Multicast and Unicast Scheduling

[0084] Figure 11A conceptually illustrates a pipelined approach for scheduling multicast and unicast traffic according to one embodiment of the present invention. Multicast scheduling 1100 is shown along the top row. Unicast scheduling 1110 is shown along the middle row. The bottom row indicates the resulting combined schedule 1120. The hollow arrows point in the direction of the time slot during which the resulting schedule is used. For example, multicast schedule M_0 is generated during time slot t_0, but is used during time slot t_1.

[0085] A multicast scheduling frequency register 1003 identifies those of the time slots, e.g., t_0 - t_7, during which multicast scheduling 1100 is to be performed. Many possible implementations of the multicast scheduling frequency register 1003 have been contemplated. According to one embodiment, a circular register is implemented, where the bits of the register are set to '1' or '0' in accordance with whether the current time slot is a multicast time slot. Each time slot, the bits of the register are rotated and the LSB or the MSB can be evaluated. Alternatively, the multicast scheduling frequency register 1003 may be implemented as an up or a down circular counter. Each time slot, the counter is incremented or decremented and the value in the counter is compared to a predetermined value that indicates when multicast scheduling is to be performed. According to one embodiment, the multicast scheduling frequency register 1003 contains a hardcoded value. In alternative embodiments, however, the multicast scheduling frequency register 1003 is a programmable parameter thereby allowing the provision of variable rate multicast servicing and putting a cap on the total bandwidth available to multicast traffic. Responsive to historical network usage or scheduled network usage, the multicast scheduling frequency register 1003 may be adjusted upward or downward automatically by a traffic monitoring process, for example. Alternatively, bandwidth may be allocated between unicast and multicast traffic by the network administrator by tuning the multicast scheduling frequency.

[0086] In the example illustrated by Figure 11A, if a bit position in the multicast scheduling frequency register 1003 contains a '1,' then multicast scheduling 1100 is performed during the corresponding time slot; otherwise no multicast scheduling is performed during the time slot. According to the embodiment depicted, unicast scheduling 1110 is performed every time slot. Assuming the duration of a time slot is insufficient to accommodate both unicast and multicast scheduling in serial, the novel combined scheduling approach described herein permits a combined schedule 1120 to be generated by employing a pipelined scheduling approach.

[0087] Briefly, in this example, multicast scheduling time slots occur as a subset of unicast scheduling time slots. Unicast scheduling 1110 occurs every time slot and multicast scheduling 1100 occurs every other time slot. During the first time slot, t_0, both multicast scheduling and unicast scheduling are performed in parallel. The multicast scheduling cycle produces a first multicast schedule, M_0, for the next time slot, t_1, and the unicast scheduling cycle produces a first unicast schedule, U_0, for the current time slot, t_0. Since no multicast cells are scheduled for t_0, the resulting combined schedule is U_0.

[0088] Referring now to the next time slot, t_1, no multicast scheduling is performed, but the results of the previous multicast scheduling cycle, M_0, are fed into the unicast scheduling cycle. As a result, only those ports that are left unclaimed by M_0 are available for use by the unicast scheduling. The resulting unicast schedule, U_1,

a predetermined threshold. For example, de-assertion of the backpressure signal may be delayed until the backpressuring output queue is half-empty.

[0094] Figure 12 is a flow diagram illustrating backpressure processing according to one embodiment of the present invention. At step 1210, the fabric arbiter 110 receives back pressure signals from the output ports 109. At step 1220, a determination is made whether or not to obey the backpressure signals based upon the size (length) of the multicast queues. If a multicast queue reaches a certain high watermark threshold, processing continues with step 1230. Otherwise, processing proceeds to step 1240.

[0095] At step 1230, requests associated with backpressuring output ports are not masked and are therefore considered eligible for multicast scheduling. In this manner, the head-of-line cell is sent to the destined output port(s) 109 regardless of the backpressure signal thereby removing the blockage and allowing the remaining multicast cells an opportunity to be scheduled for transfer across the fabric 120. At step 1240, no multicast build up has occurred, therefore, requests associated with backpressuring output ports are masked and are not considered by the multicast scheduler 215. After either of steps 1230 or 1240, processing continues with step 1250. At step 1250, multicast scheduling is performed based upon the requests that remain after step 1230 or 1240. Advantageously, one overloaded output port is prevented from adversely affecting other unrelated output ports.

[0096] Referring now to Figure 13, a block diagram of an exemplary hardware implementation of a portion of backpressure logic 1300 will now briefly be described, according to one embodiment of the present invention. An original request vector 130 is masked or not based upon a backpressure override signal 1330 and a resulting request vector 1340 is output from the backpressure logic 1300. In this example, the backpressure logic 1300 comprises a backpressure mask 1305, N AND gates 1320, and N multiplexors 1325. In this example, a bit position of the backpressure mask 1305 contains a '0' if the corresponding output port / queue is backpressuring or a '1' is the output port /queue is not backpressuring. In this manner, the requests associated with backpressured output ports / queues can be masked off. However, the backpressure override signals 1330 allow one or more of the original request vector positions to be passed through unchanged in the case of multicast queue buildup at an input port, for example.

[0097] In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

Claims

1. A method of scheduling multicast data for transfer across a fabric of a network device, the method comprising the steps of:

   a scheduler receiving a transmit request associated with each of a plurality of input ports, the transmit request identifies those of a plurality of output ports to which pending multicast cells are ready to be transmitted, if any; and
   for each of a plurality of classes of service, the scheduler performing a single scheduling iteration including
   granting one or more of the plurality of input ports access to the fabric by issuing grants for each of the one or more of the plurality of input ports based upon the transmit requests and a priority indicator that identifies an input port of the plurality of input ports that is given scheduling priority for the single scheduling iteration,
   on behalf of each of the plurality of input ports, accepting all grants corresponding to the input port, and
   updating the priority indicator for use in a subsequent scheduling cycle.

2. The method of claim 1, further comprising the step of communicating a configuration of the fabric to the input ports and activating the fabric, and wherein the step of updating the priority indicator for use in a subsequent scheduling cycle comprises updating the priority indicator if the transmit requests indicate no multicast data is ready for transfer across the fabric or if the configuration will completely transfer a multicast cell associated with the input port identified by the priority indicator to all of the output ports to which it is destined.

3. The method of claim 1 or 2, the method further comprising the steps of maintaining only a single priority indicator for each class of service, and wherein the scheduling iterations are performed in decreasing order of the relative priorities of the plurality of classes of service.

4. The method of claim 1, 2 or 3, wherein the priority indicator comprises a circular counter having N states, and wherein the step of updating the priority indicator for use in a subsequent scheduling cycle comprises rotating to a next input port of the plurality of input ports in a round robin fashion.

5. A method of performing a scheduling iteration to scheduling multicast data for transfer across a fabric of a network device, the method comprising the steps of: during a first scheduling phase of a current scheduling iteration, a scheduler receiving a trans-

request associated with the input port identified by the priority indicator and the configuration of the fabric generated during the current cell cycle.

13. A method of forwarding multicast data in a network device, the method comprising the steps of:

a. during a current cell scheduling cycle, receiving a transmit request associated with each of a plurality of input ports that has a multicast cell ready for transmission across a fabric to one or more of a plurality of output ports, the transmit request identifying those of the plurality of output ports to which the multicast cell is destined;
b. scheduling multicast cells for transmission across the fabric by communicating a configuration of the fabric for the current cell scheduling cycle to the plurality of input ports and the fabric, the configuration based upon the transmit requests and a round robin indicator that identifies an input port of the plurality of input ports that is given scheduling priority for the current cell scheduling cycle; and;
c. updating the round robin indicator for use in a next cell scheduling cycle based upon completion of cell transmission from the input port identified by the round robin indicator to all of the output ports to which it is destined.

14. The method of claim 13, wherein each input port of the plurality of input ports includes a multicast queue for each of a plurality of classes of service, wherein a separate priority indicator is maintained for each class of service, and wherein the method comprises repeating steps (a)-(c) sequentially for each of the plurality of classes of service in decreasing order of priority.

15. The method of claim 14, further comprising the step of maintaining an availability indicator for each of the plurality of input ports and each of the plurality of output ports, and wherein the configuration is further based upon the availability indicators.

16. A method of forwarding multicast data in a network device, the method comprising the steps of:

during a current cell scheduling cycle, receiving a transmit request associated with each of a plurality of input ports that has multicast cells ready to be transmitted across a fabric to one or more of a plurality of output ports, the transmit request identifying those of the plurality of output ports to which a multicast cell at the head of the input port's multicast queue is destined;
maintaining a round robin counter for each of a plurality of classes of service, each of the round robin counters identifying an input port of the plurality of input ports that is given scheduling priority for the corresponding class of service during the current cell scheduling cycle;
generating a configuration of the fabric based upon the transmit requests and the round robin counters;
scheduling multicast cells for transmission across the fabric by communicating the configuration of the fabric to the plurality of input ports and the fabric; and
updating the round robin counters for use in a next cell scheduling cycle based upon a set of conditions that are always true for the highest priority level of the plurality of classes of service.

17. A network device comprising:

a plurality of input ports and output ports residing on a plurality of line cards, the plurality of output ports;
a fabric coupled to the plurality of line cards, the fabric including a switched backplane allowing the simultaneous transfer of data among the plurality of input ports and output ports; and
a multicast scheduler coupled to the fabric and the plurality of line cards, the multicast scheduler configured to:
receive a transmit request associated one or more of the plurality of input ports indicating multicast data that is ready to be transmitted across the fabric if any, the transmit request identifying those of the plurality of output ports to which multicast data is destined
maintain a round robin counter for each of a plurality of classes of service, each of the round robin counters identifying an input port of the plurality of input ports that is given scheduling priority for the corresponding class of service during a current cell scheduling cycle,
generate a fabric configuration based upon the transmit requests and the round robin counters,
schedule multicast cells for transmission across the fabric by communicating the configuration of the fabric to the plurality of input ports and the fabric, and
update the round robin counters for use in a next cell scheduling cycle based upon a set of conditions that are always true for the highest priority level of the plurality of classes of service.

18. A computer program element comprising computer

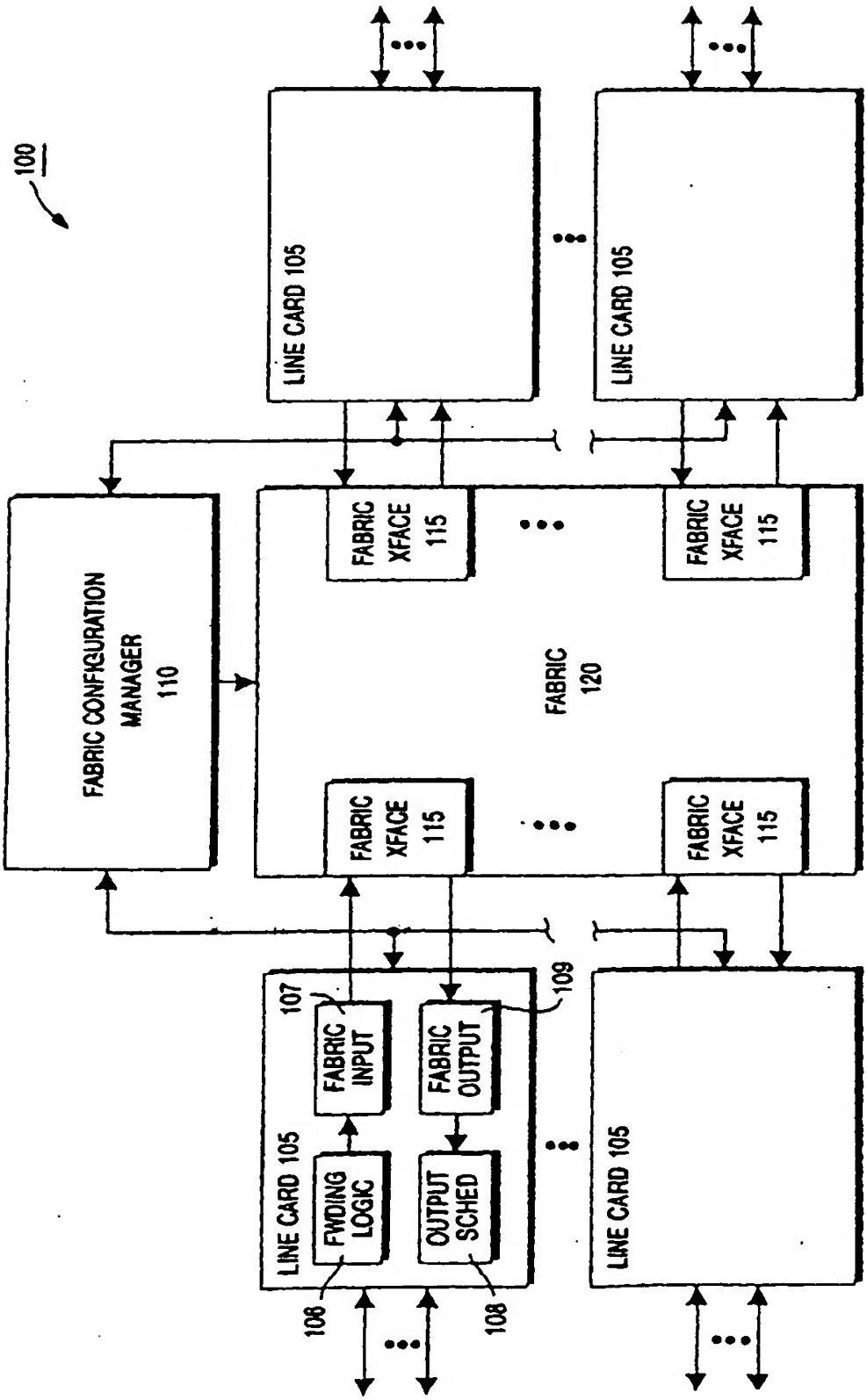

FIG. 1

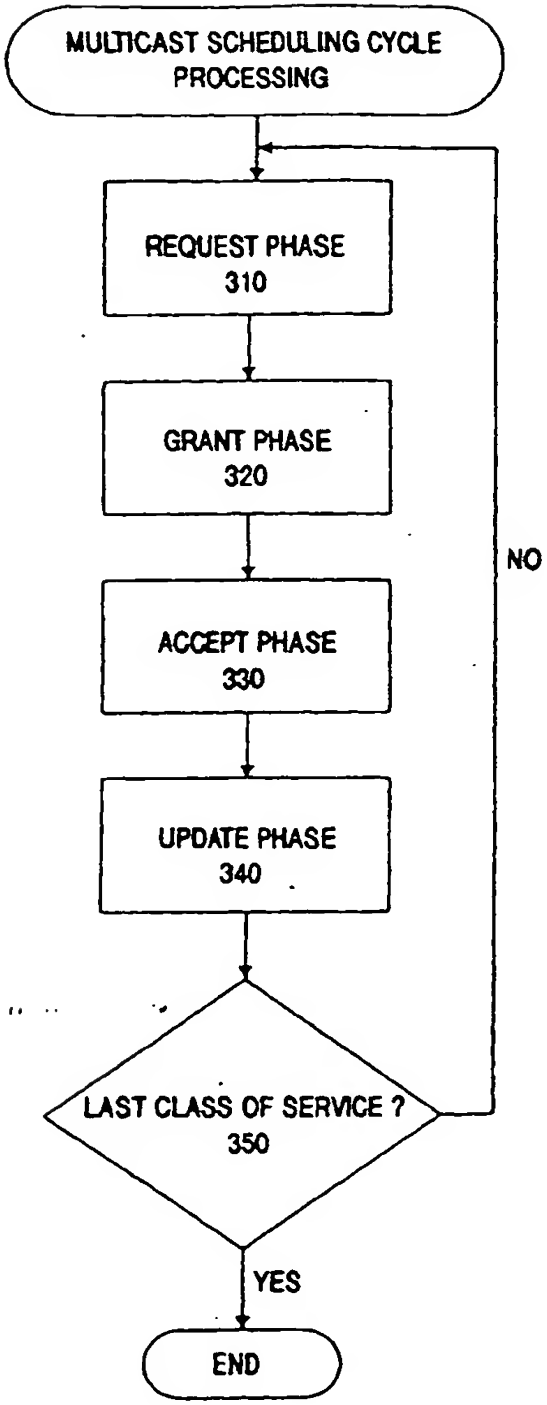

FIG. 3

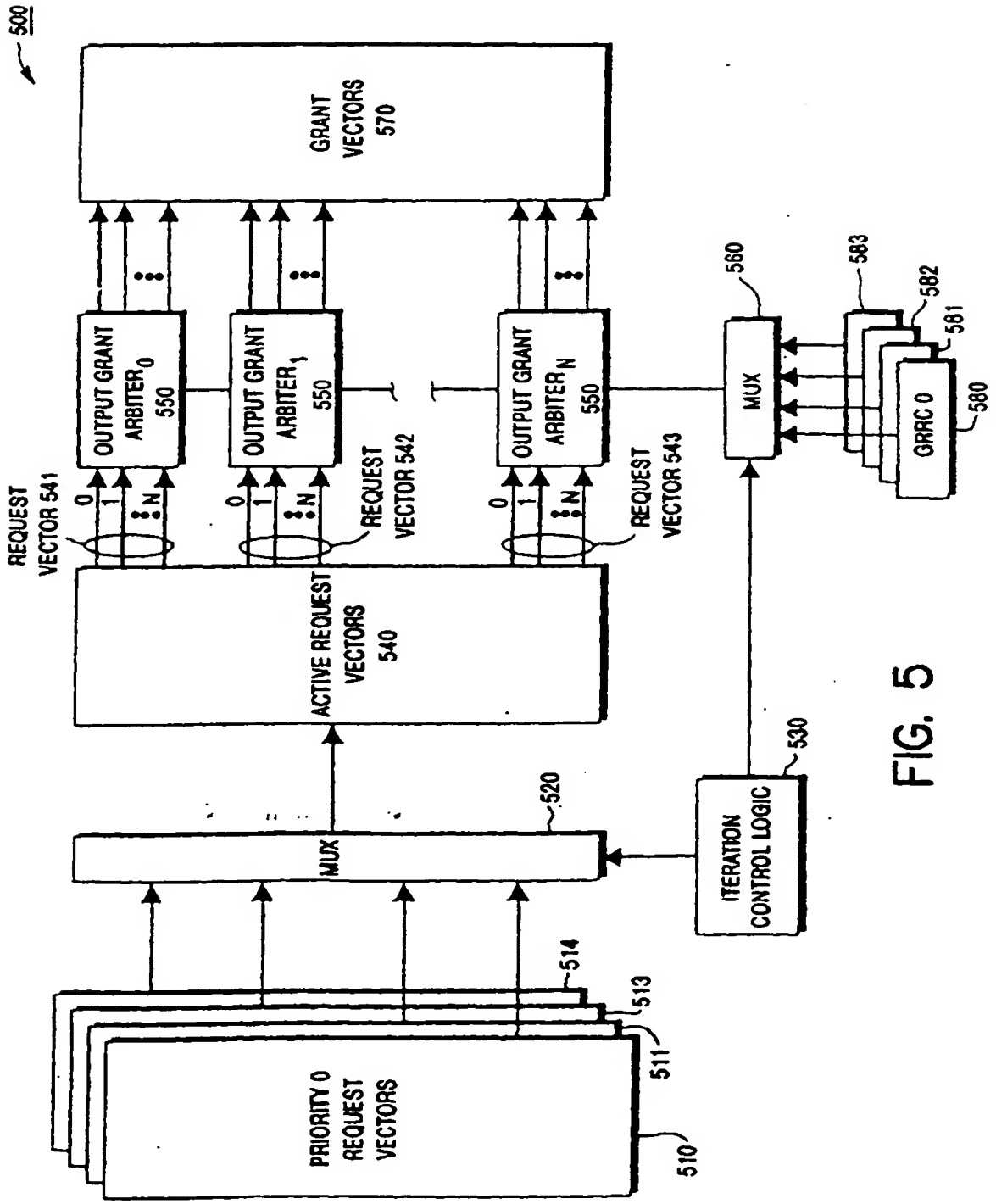

FIG. 5

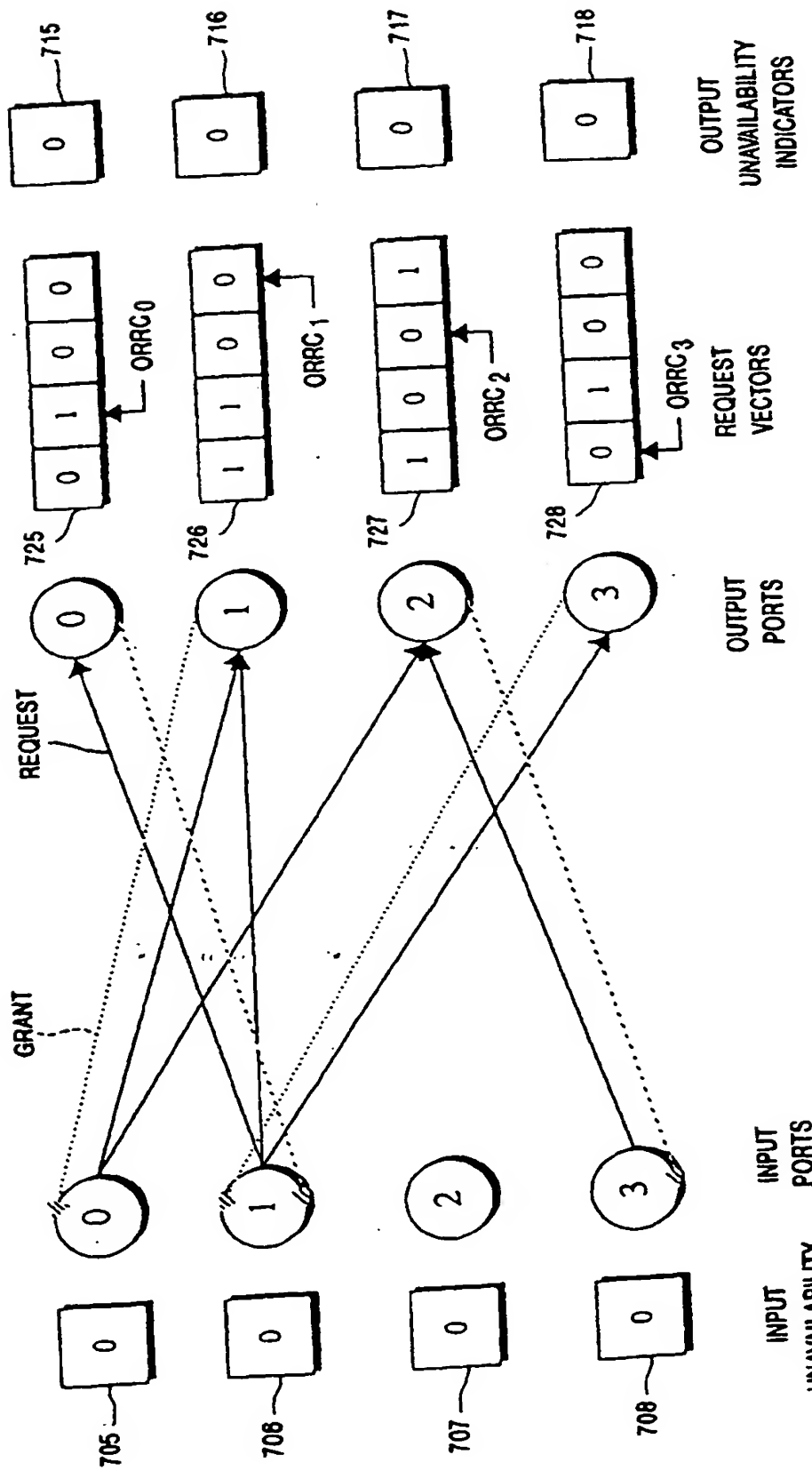

FIG. 7A

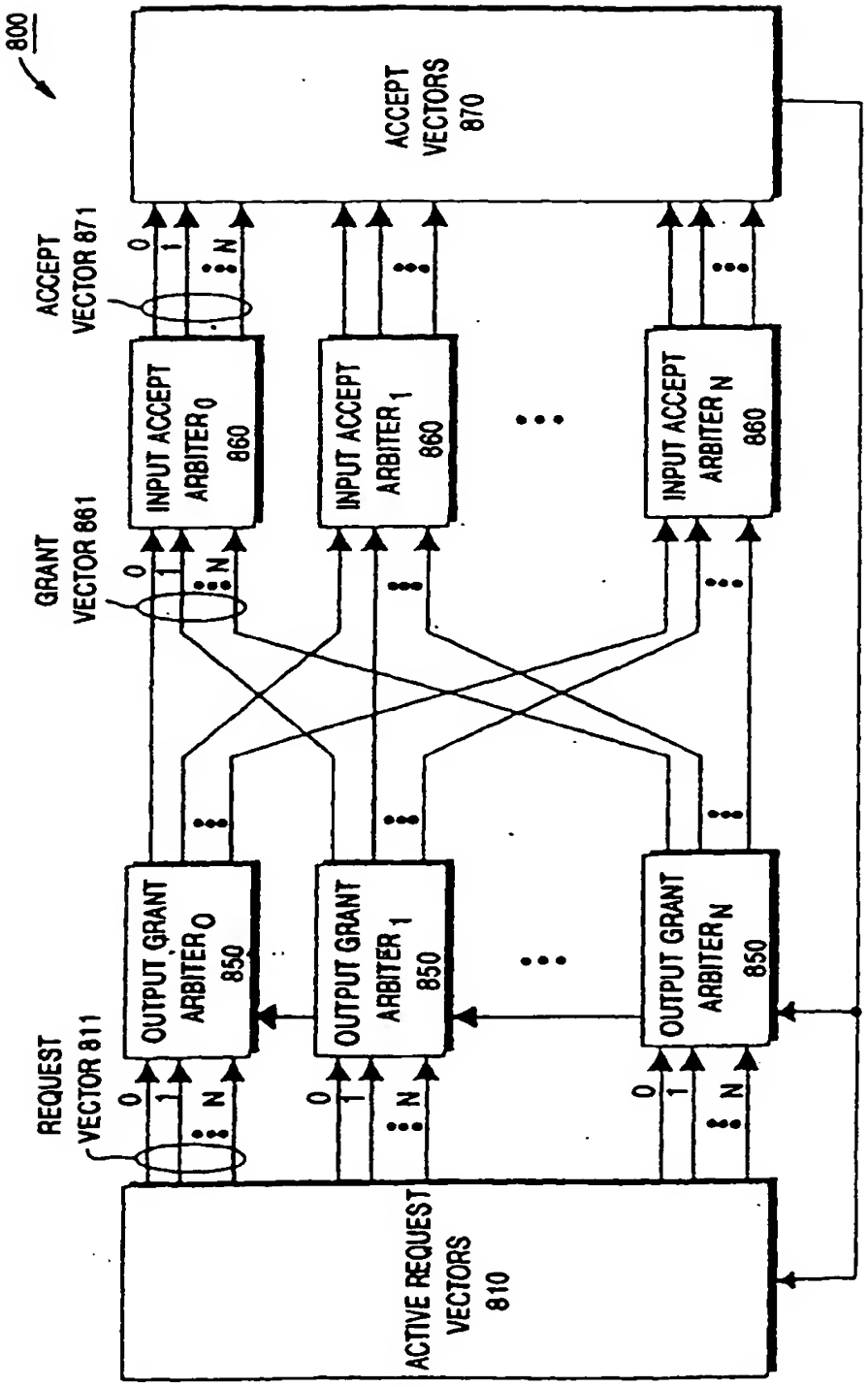

FIG. 8

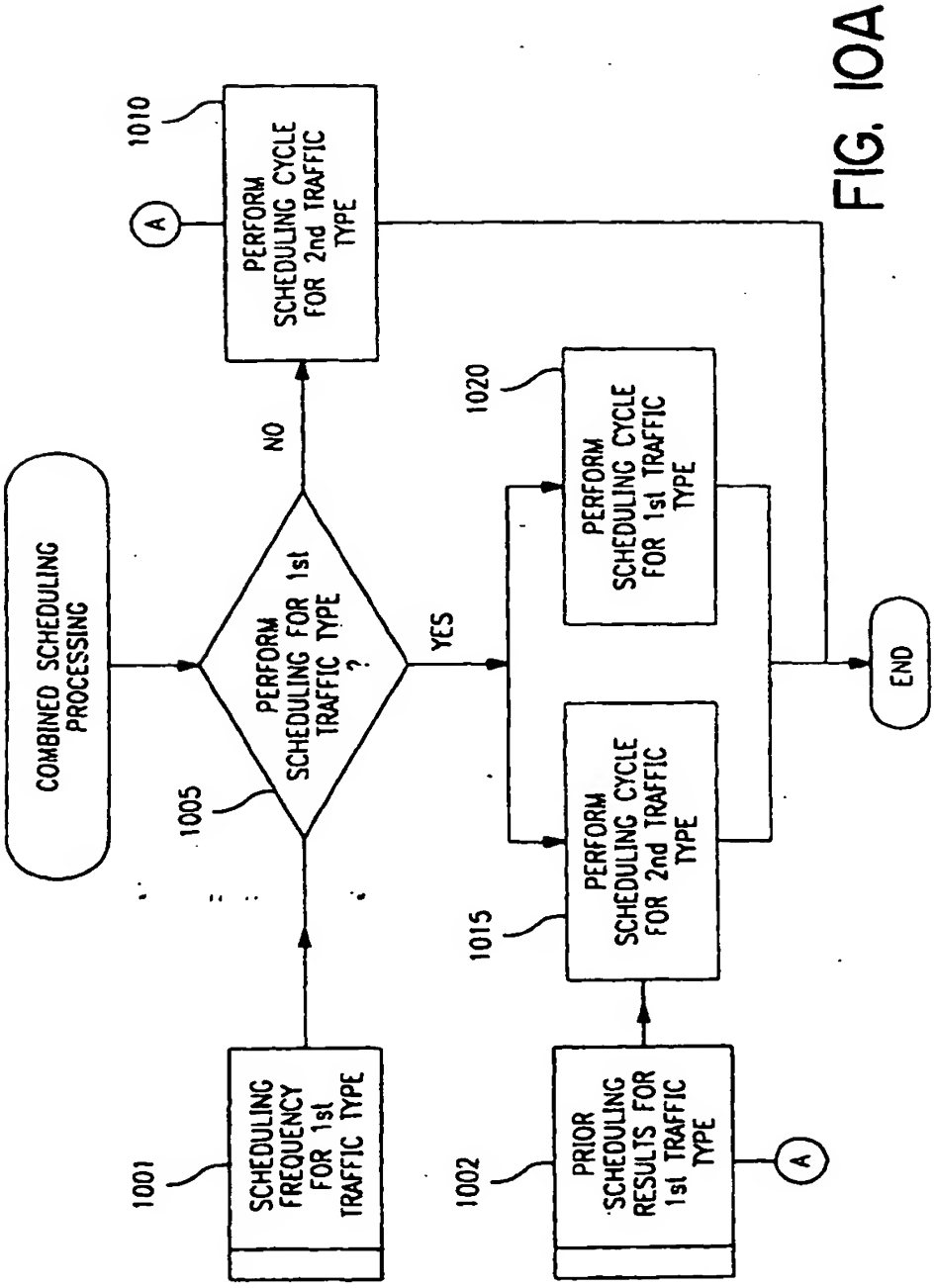

FIG. 10A

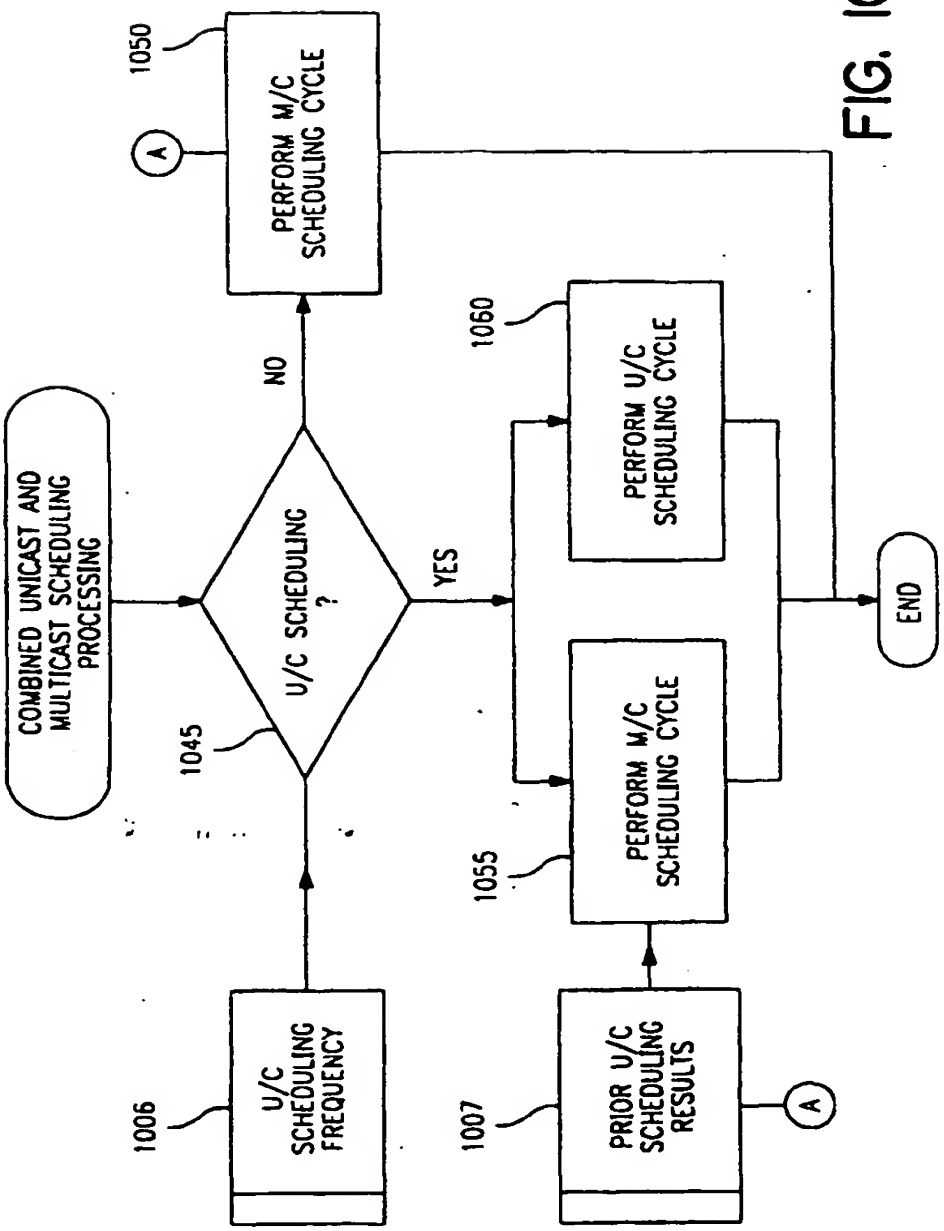

FIG. 10C

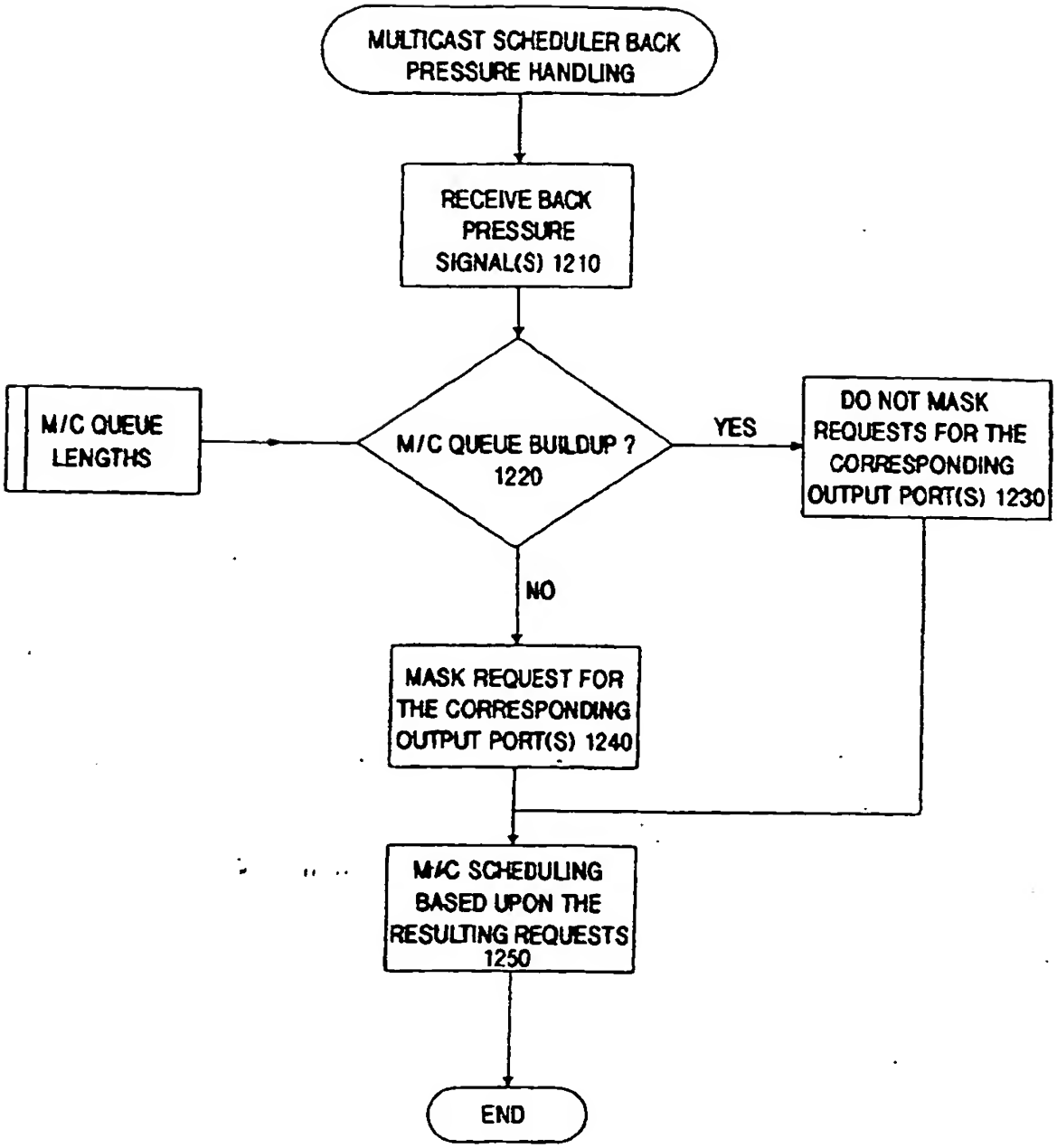

FIG. 12

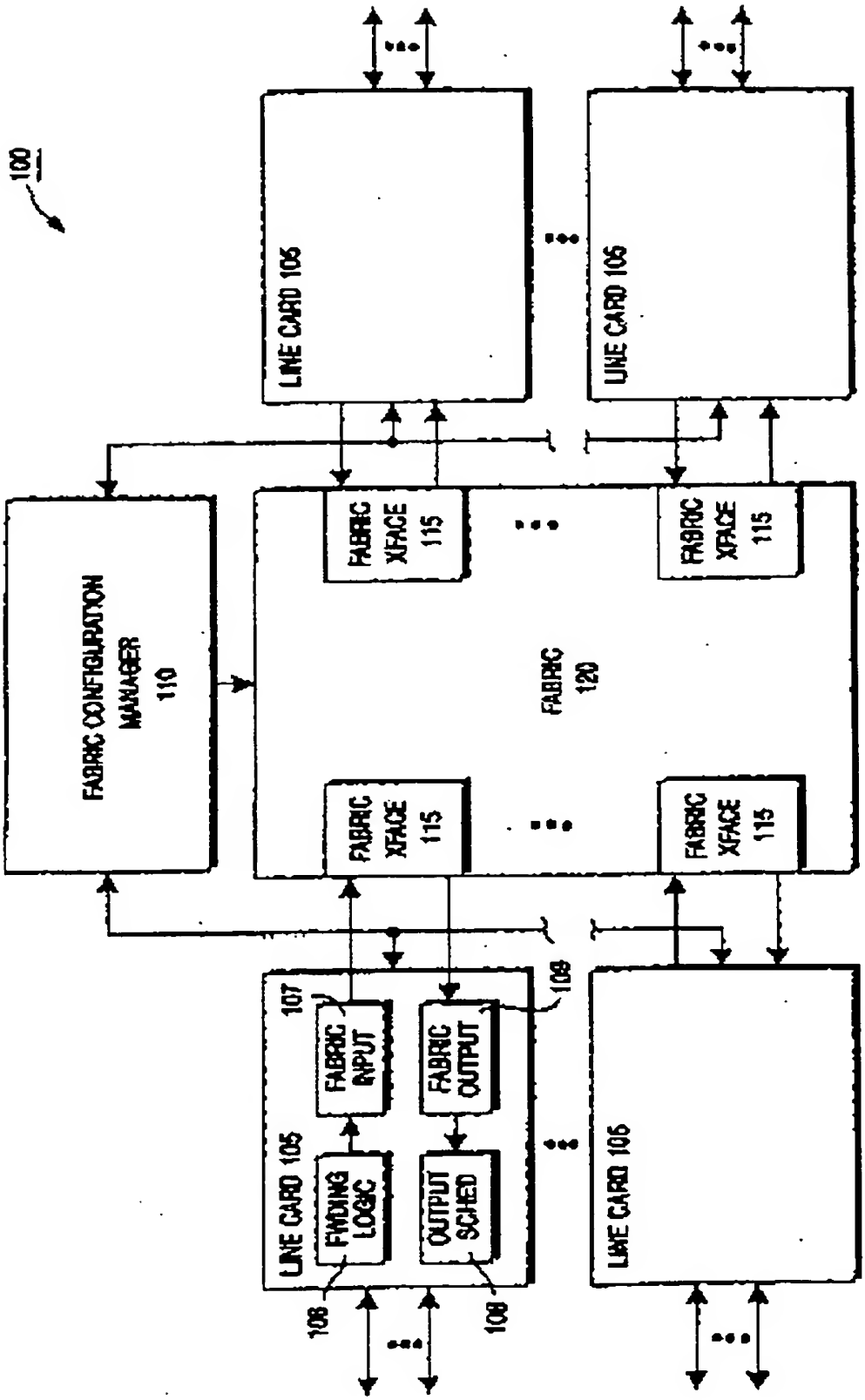

FIG. 1

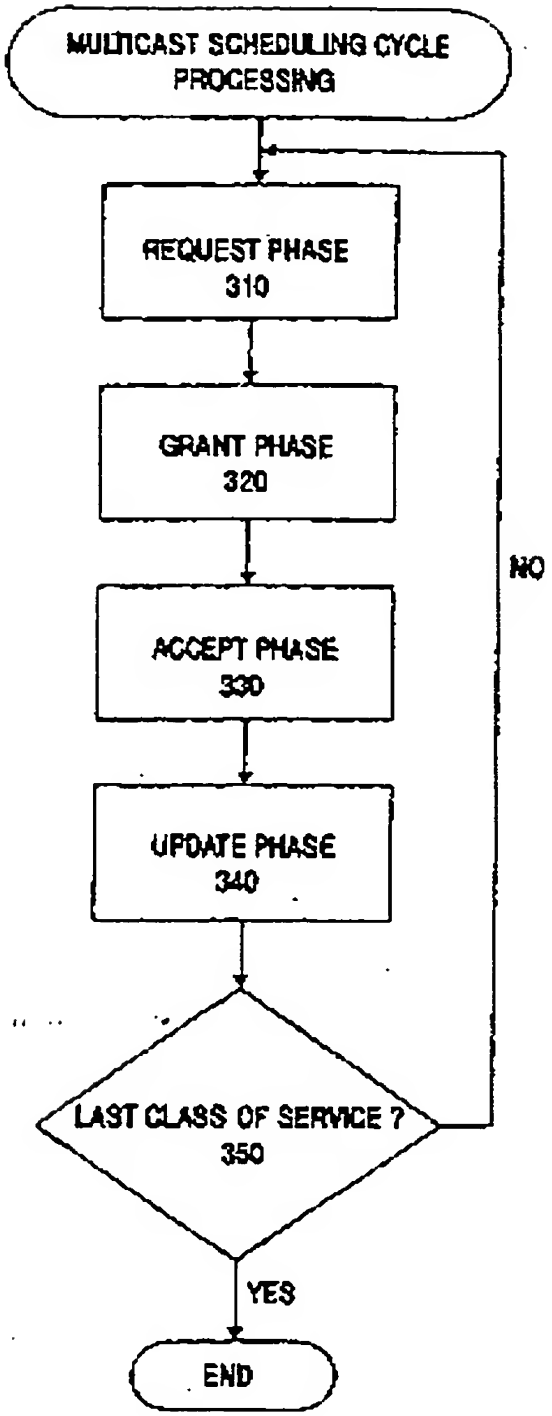

FIG. 3

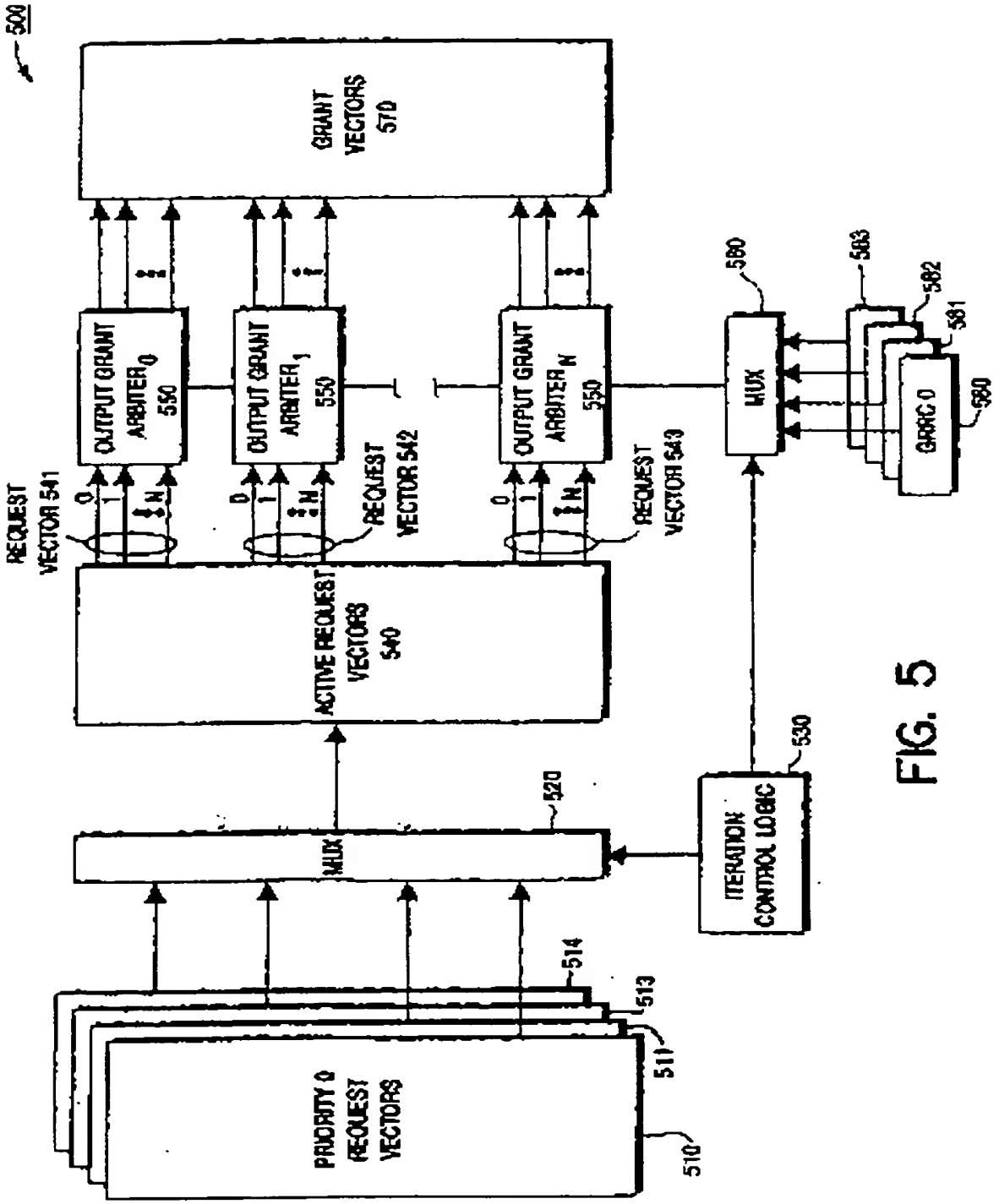

FIG. 5

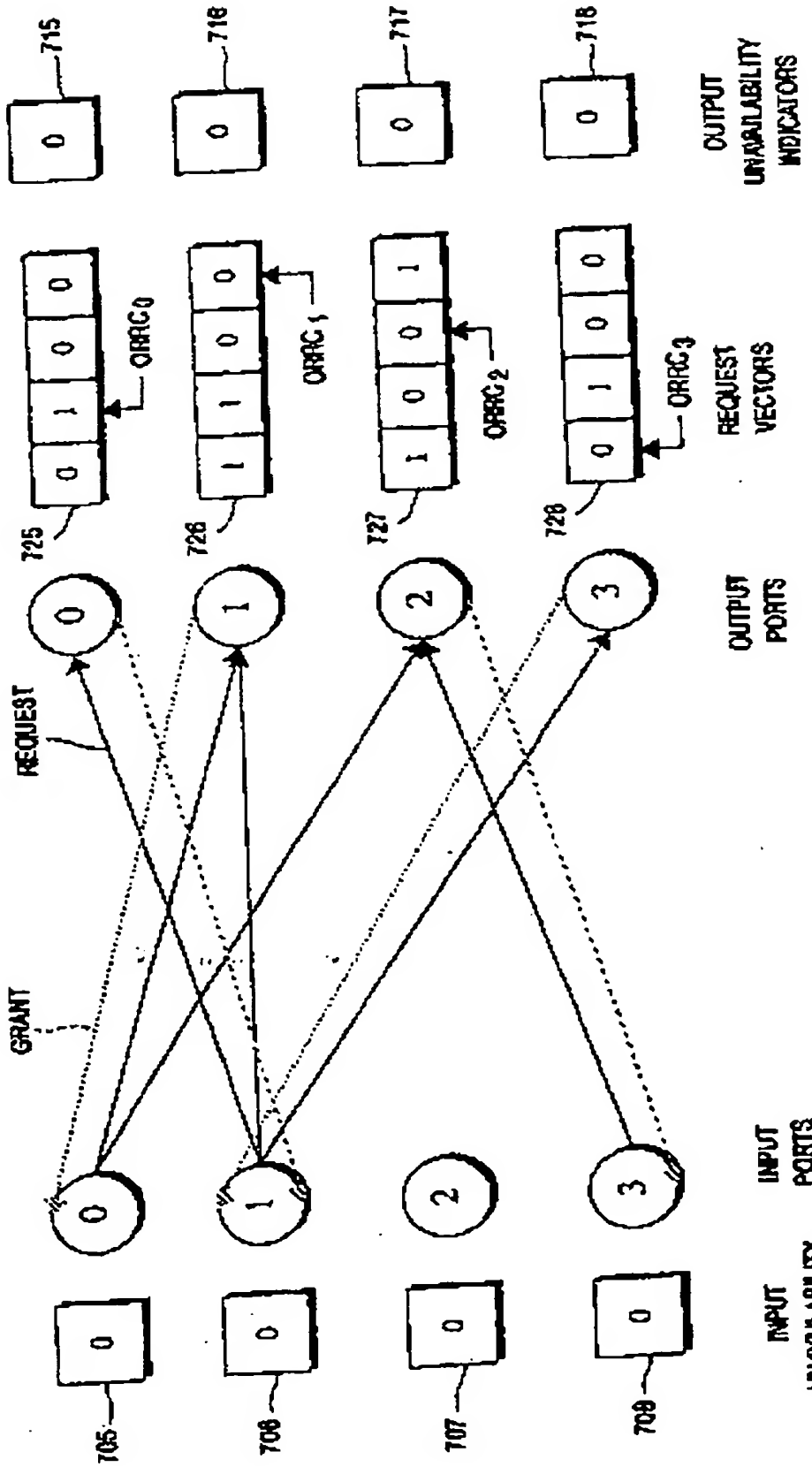

FIG. 7A

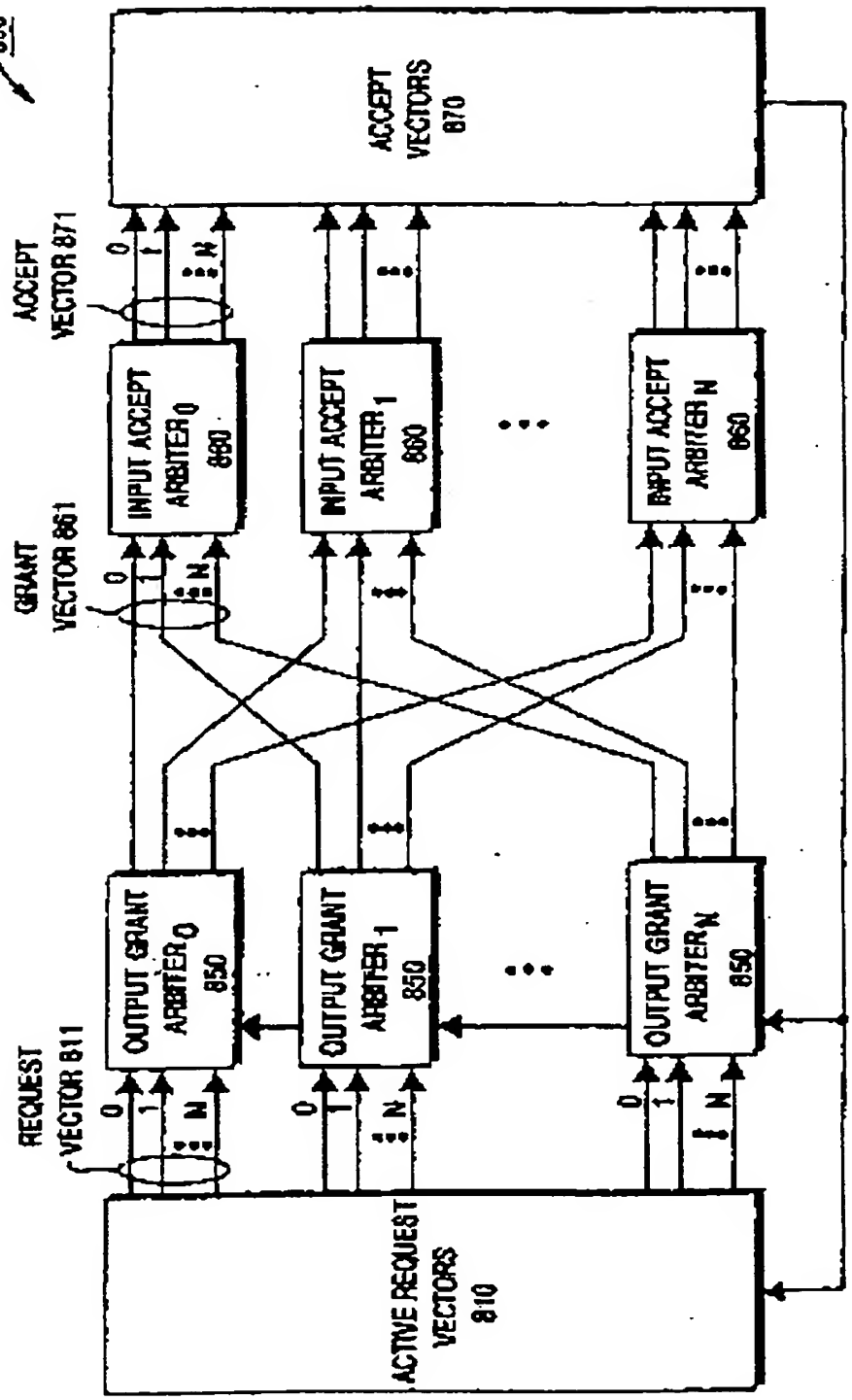

FIG. 8

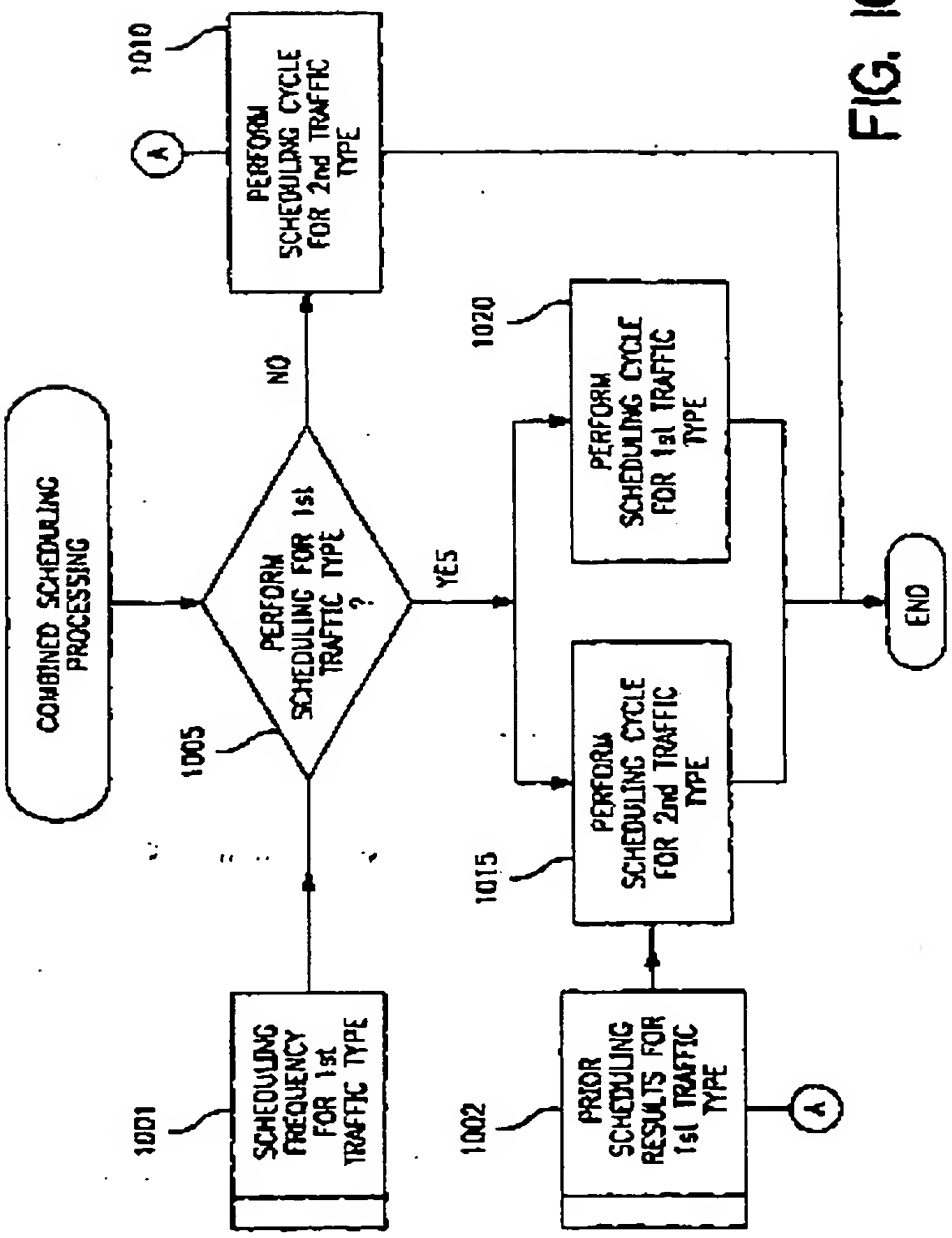

FIG. 10A

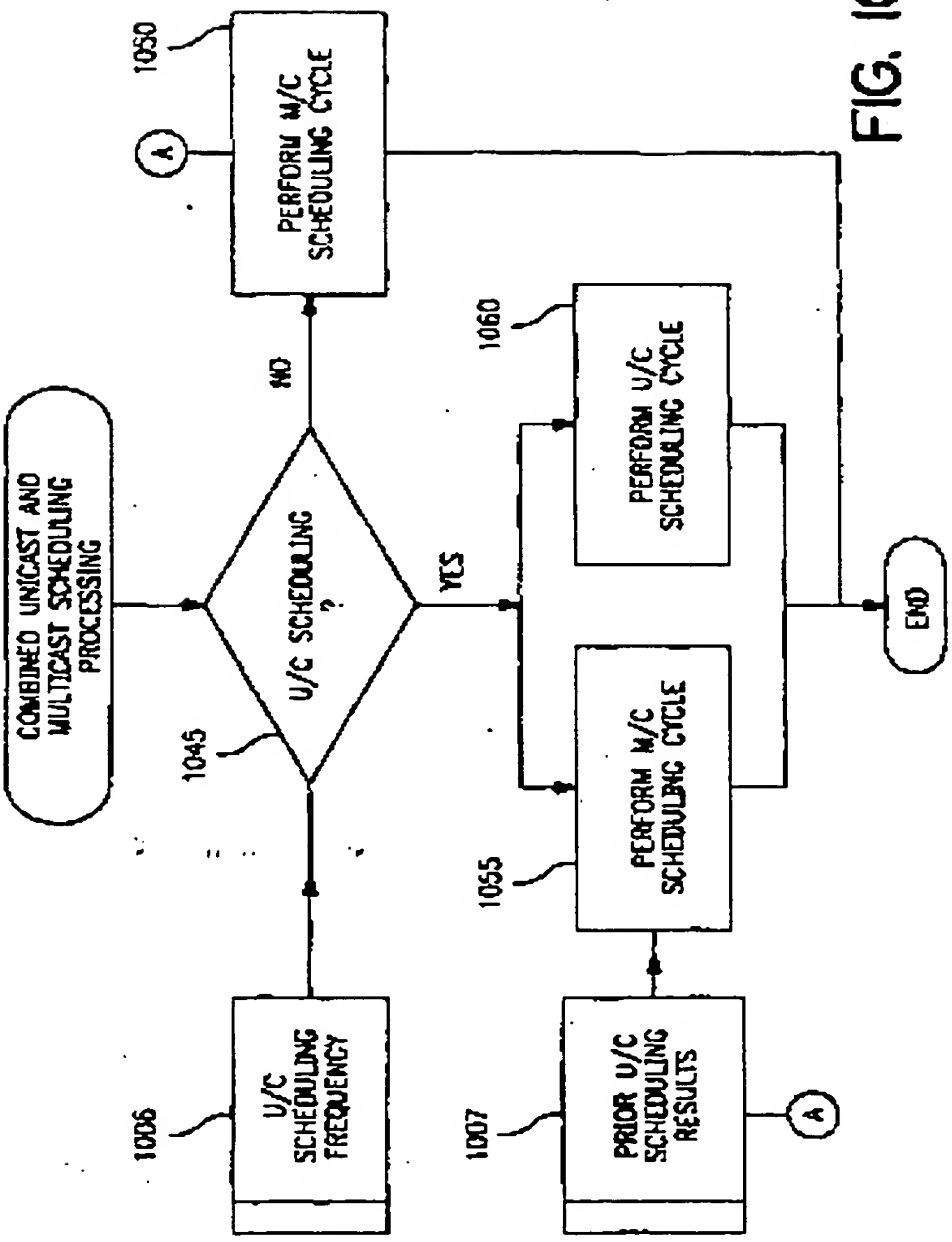

FIG. 10C

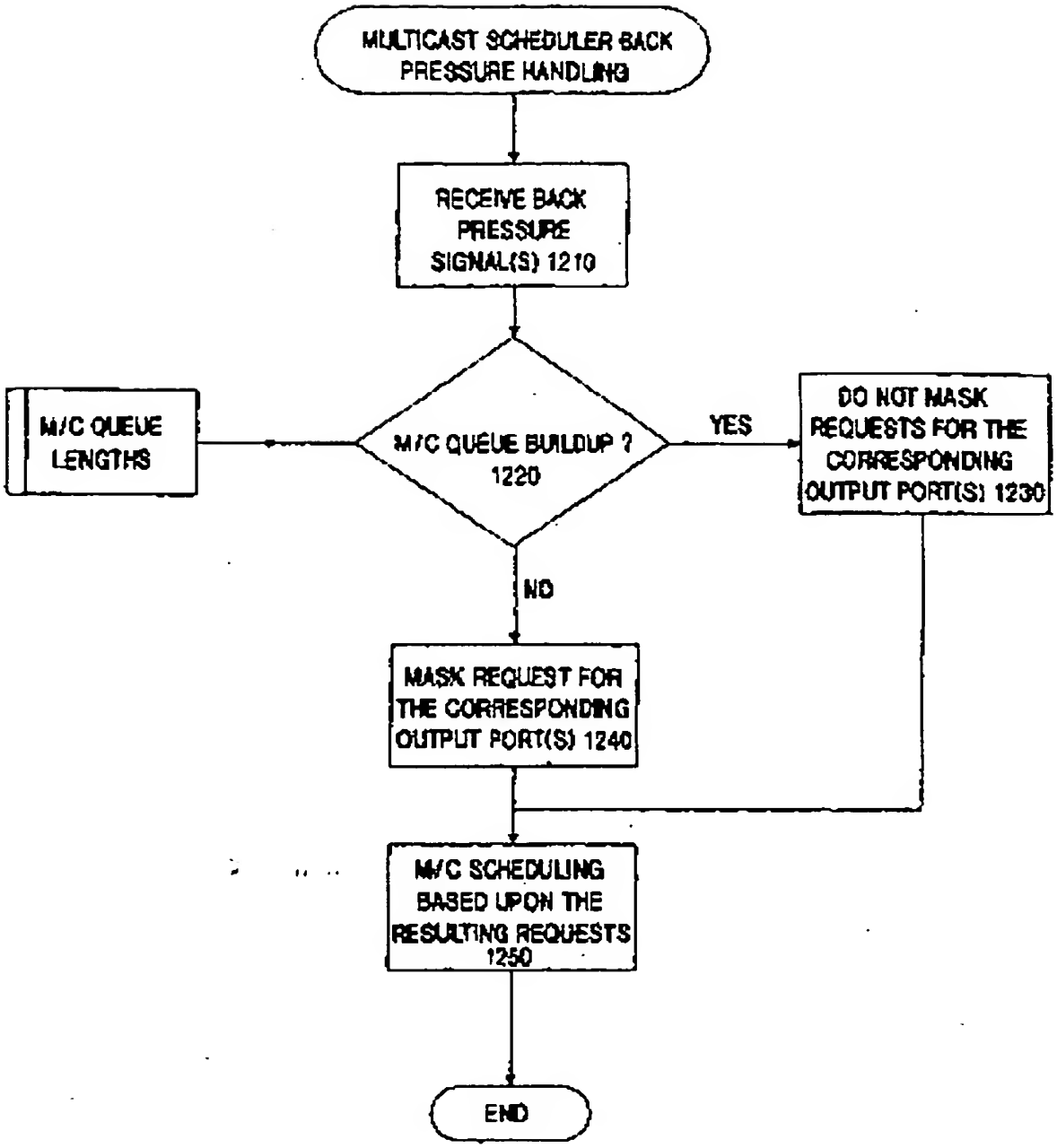

FIG. 12